TM No. 94-1004

NAVAL UNDERSEA WARFARE CENTER
DETACHMENT, NEW LONDON CONNECTICUT

Technical Memorandum

A PERFORMANCE ANALYSIS OF COMMUNICATIONS BETWEEN TWO MASSIVELY PARALLEL COMPUTERS: THE CONNECTION MACHINE CM-200a AND THE 64-CELL iWarp

Date: 12 January 1994

Prepared by: Matthew J. Krzych
Matthew J. Krzych
Systems Architecture &
Software Development Branch
Submarine Sonar Department

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
12 JAN 1994	**Technical Memorandum**	**12-01-1994 to 12-01-1994**

4. TITLE AND SUBTITLE
A Performance Analysis of Communications between Two Massively Parallel Computers : the Connection Machine CM-200a and the 64-Cell iWarp

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Matthew Krzych

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Undersea Warfare Center Division,Newport,RI,02841

8. PERFORMING ORGANIZATION REPORT NUMBER
TM 941004

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research, Computer Technology Block

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
NUWC2015

14. ABSTRACT
Two massively parallel computers consisting of Thinking Machines Corp's Connection Machine CM-200a and Intel Corp's 64-cell iWarp are being used within an experimental sonar subsystem to evaluate their impact on various signal processing algorithms. Each machine performs specific tasks within the sonar processing thread and are required to communicate with one another in a tightly coupled manner. This interface is in the process of being defined. A number of design options exist; however, it is not obvious which of the options would provide optimum performance. As a result, several studies are being conducted to help resolve this issue. This paper documents two such studies, one which focuses on the performance of various I/O command suites available on the Connection Machine, the other on the performance of reformatting data (corner turning) so that data generated on one machine (the iWarp) is usable by the other (the Connection Machine).

15. SUBJECT TERMS
Connection Machine; iWarp

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**16**	

ABSTRACT

Two massively parallel computers consisting of Thinking Machines Corp's *Connection Machine CM- 200a* and Intel Corp's *64-cell iWarp* are being used within an experimental sonar subsystem to evaluate their impact on various signal processing algorithms. Each machine performs specific tasks within the sonar processing thread and are required to communicate with one another in a tightly coupled manner. This interface is in the process of being defined. A number of design options exist; however, it is not obvious which of the options would provide optimum performance. As a result, several studies are being conducted to help resolve this issue. This paper documents two such studies, one which focuses on the performance of various I/O command suites available on the Connection Machine, the other on the performance of reformatting data (corner turning) so that data generated on one machine (the iWarp) is usable by the other (the Connection Machine).

Of the various I/O command suites available on the Connection Machine, it was found that all provide relatively the same performance and throughput capacities. This is contrary to initial predictions where it was anticipated that streaming I/O and parallel reads would provide significantly better performance. The results of this study suggest there exists a great deal of freedom available with the Connection Machine I/O thereby placing any design restrictions on other factors.

The corner turning study concluded the Connection Machine is more than capable of performing the data reformatting process particularly when the amount of data to be corner turned is kept under 4 MBytes in size. Based on more lenient ASPA timing requirements, this number can be extended to 32 MBytes. It was also found that the time it takes to corner turn data is dependent on the total amount of data processed and is affected little by the number of elements along a dimension.

Other studies need to be conducted in order to completely specify the iWarp to Connection Machine interface. Due to the unavailability of the 64-cell iWarp at the time of these studies, iWarp I/O was not investigated. It can be anticipated that this will have a major impact on communications between the two machines.

ADMINISTRATIVE INFORMATION

This Memorandum was prepared under the Advanced Sonar Processing Architectures (ASPA) program, Principal Investigator José Muñoz. The sponsoring activity is the Office of Naval Research (ONR), Computer Technology Block, Program Manager Elizabeth Wald.

1.0 INTRODUCTION

The Advanced Sonar Processing Architectures (ASPA) project is currently tasked with investigating the applicability of using several different types of high performance computers for sonar processing, particularly within the scope of long-line towed arrays. These computers include Thinking Machines Corp.'s *Connection Machine CM-200a* (CM) and Intel Corp's *64-cell iWarp*, both of which are massively parallel computers. An experimental sonar subsystem is in the process of being developed in which these computers represent an integral part of the system. This subsystem provides an "arrays-to-displays" capability including processing for beamforming, detection, tracking, and display generation.

Processing within this system has been allocated such that the iWarp performs front-end tasks of the sonar processing thread, namely beamforming, while the CM performs much of the remaining tasks. Data flows through the system in a pipeline fashion with initial processing performed by the iWarp which, when complete, is forwarded to the CM. Due to the enormity of the data involved as well as stringent timing requirements, a tight, efficient interface is required between the iWarp and the CM-200a. This interface is of special interest since two massively parallel computers are required to communicate with one another, each employing different architectural paradigms; the iWarp represents a Multiple Instruction Multiple Data (MIMD) machine while the CM-200a a Single Instruction Multiple Data (SIMD). For both machines, I/O capacities represent a significant limitation and potential bottleneck. This paper documents a study conducted to help determine the optimum design for an interface between these two machines. The study examines various I/O task allocations as well as data organization.

2.0 OVERVIEW

The physical topology of the ASPA system is provided in Figure 1. The type and quantity of computers used in addition to their physical connection were specified prior to the start of the study and define the boundaries within which the study was to be conducted. Given this architecture, the study's primary focus was to examine various I/O task allocations, read operations, and data transfer layouts so as to maximize throughput when sending data from the iWarp to the CM. The study did not investigate adding additional computing resources or communication channels. At the time of this study, the 64-cell iWarp was physically removed from the system and not readily available. As a result, iWarp timing results were not obtained; iWarp functionality was simulated to the maximum extent possible.

Given the system in Figure 1, sensor data enters the system via a Sensor host computer (a Sun 4 workstation) where it is forwarded to the iWarp for beamform processing; the beamformed data is then placed out on the VME bus via a Sun Interface Board (SIB). Once on the VME bus, the data can be buffered in several different locations before it is passed to the Connection Machine; potential buffers include 1) a data vault on the VME host (a Sun 4/360 workstation), 2) VME memory, or 3) a dedicated disk on the Connection Machine host (a Sun 4/670MP workstation). Both the data vault and VME memory can be accessed directly via VME, access to the CM host disk requires going through the VME host to Ethernet to the CM host. Once at the Connection Machine, data is processed using various detection and tracking algorithms; data is also pre-processed for display generation. The CM data is then forwarded to the display device via the CM host.

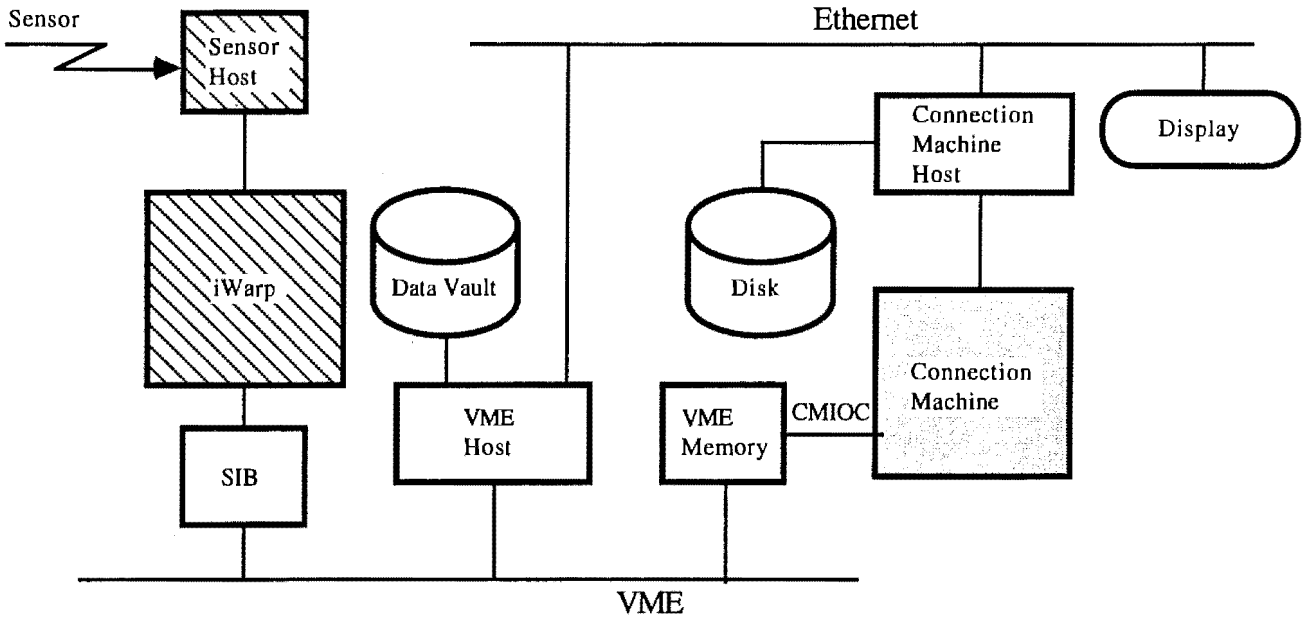


Figure 1 - System Overview

As mentioned above, the data that is passed from the iWarp to the Connection Machine has been beamformed and resides in frequency-beam space. A pictorial representation is presented in Figure 2. Its exact size (# frequency bins vs # beams vs # time steps) has yet to be specified and will be dependent upon the performance of various algorithms running on the iWarp and CM as well as the iWarp/CM interface. It should be noted that due to CM architecture restrictions, data must be organized and stored such that the number of elements along any of its dimensions is a power of two (eg. 64, 128, 256, etc.). In addition, the total number of elements within a data block must exceed 8192. If the actual data does not meet either of these requirements, the assigned data structure must be over-sized. As an example, 51 beams of actual data must be stored along a dimension containing 64 beams. This has the potential of making the CM extremely inefficient particularly when there is a large discrepancy between actual and declared data sizes.

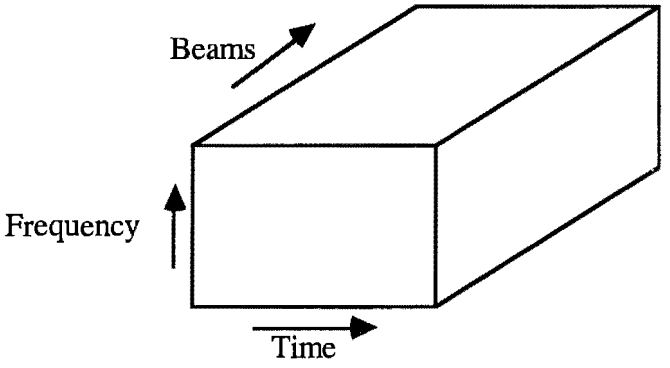


Figure 2 - Data Layout

The remainder of this report describes two independent studies conducted to address some of the design issues associated with the iWarp/CM interface. This includes timing various CM library calls, functions, and commands associated with each of the buffer transfer options described above. These timing results will later be used in designing an interface. The second study examines the impact that data size has on the performance of the interface. The number of frequency bins, beams, and time steps were varied so as to evaluate their role in affecting throughput. Due to architectural differences between the iWarp and CM, each machine specifies optimal ways in which data is to be laid out for processing. These data layouts differ and must be accounted for when transferring data between machines. The impact of conversion (corner turning) is considered.

3.0 TIMING STUDY OF DATA TRANSFER PROCESSES

The topology of the ASPA system provides alternative ways of transferring data from the iWarp to the CM: via a data vault, VME memory, or dedicated disk on the CM host. Each of these data flows, as highlighted in Figures 3a-c, employ different processes for transferring data. This study examines the performance of these processes so as to aid in defining the interface. It should be noted that the VME data flow analysis has yet to be performed (Figure 3b). As for the other two methods, analysis focused on getting data into the CM and not on getting the data from the iWarp to an intermediate storage location. The solid arrows in Figures 3a-c represent the data flows modeled as part of this study; the dashed arrows will be modeled at a later date.

From a CM point of view, the most trivial method of getting data into its memory is via a read from the dedicated disk on the CM host (Figure 3c). This method simply requires a read statement, placing data into CM host memory, followed by a "CM_write_to_news_array" command. The latter converts the data from a scalar format in the CM host to a parallel one so as to exploit the parallelism of the CM. Before the read takes place, the data exists on disk within a Unix file in scalar format. The disk itself is a Unix supported file system.

The alternative of storing data on disk utilizes a data vault, a CM supported file system (Figure 3a). The data vault permits one to store data in either serial or parallel format and to read the data in one of a variety of ways including synchronously, asynchronously, or buffered. When reading serial data, the data must be converted to a parallel format before it can be processed which is accomplished using either a "CMFS_transpose_always" or "CMFS_twuffle_from_serial_order_always" command. Although both commands produce the same result, they utilize different communication paths within the CM. In addition to parallelizing the data, serial data may also have to be converted to account for byte ordering anomalies. Depending on the type of machine which generated the data, the byte ordering of the data may not be compatible with that of the CM. This is the case for this study where the data was generated on a Sun workstation which uses left-to-right byte ordering ("big-endian") while the CM uses right-to-left ordering ("little-endian"). The "CMFS_cm_to_standard_byte_order" command performs this conversion.

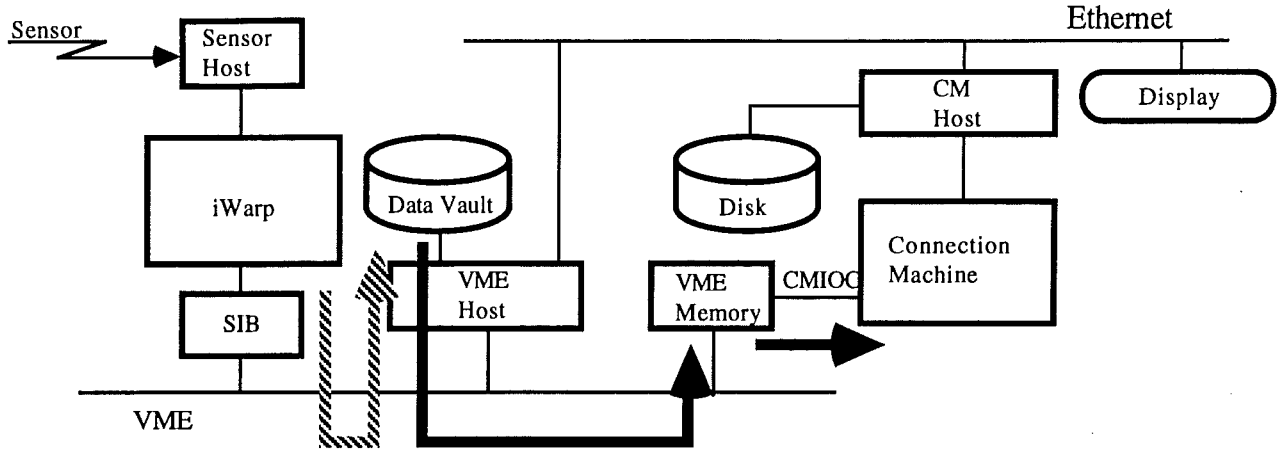


Figure 3a - Data Flow via Data Vault

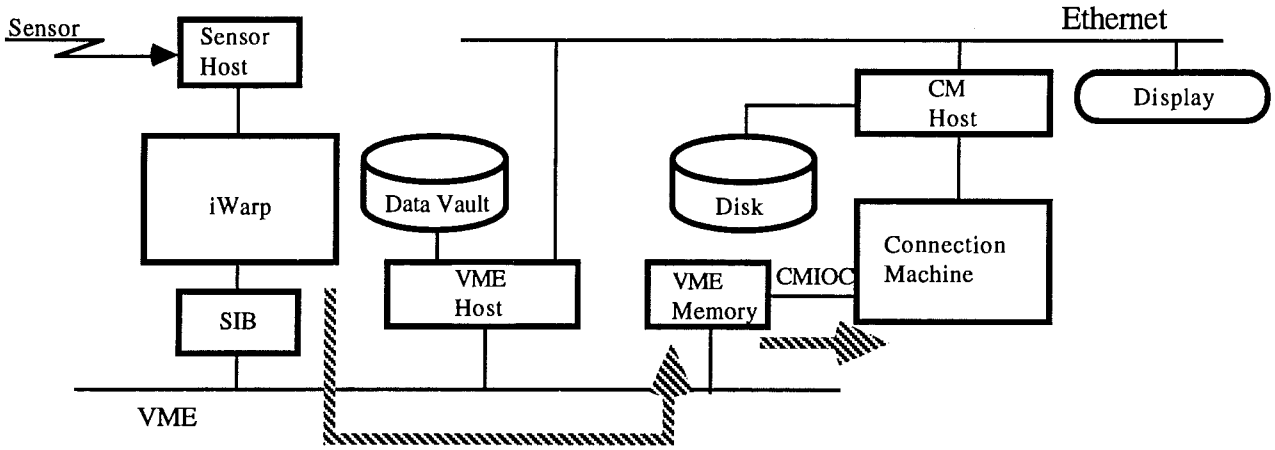


Figure 3b - Data Flow via VME Memory

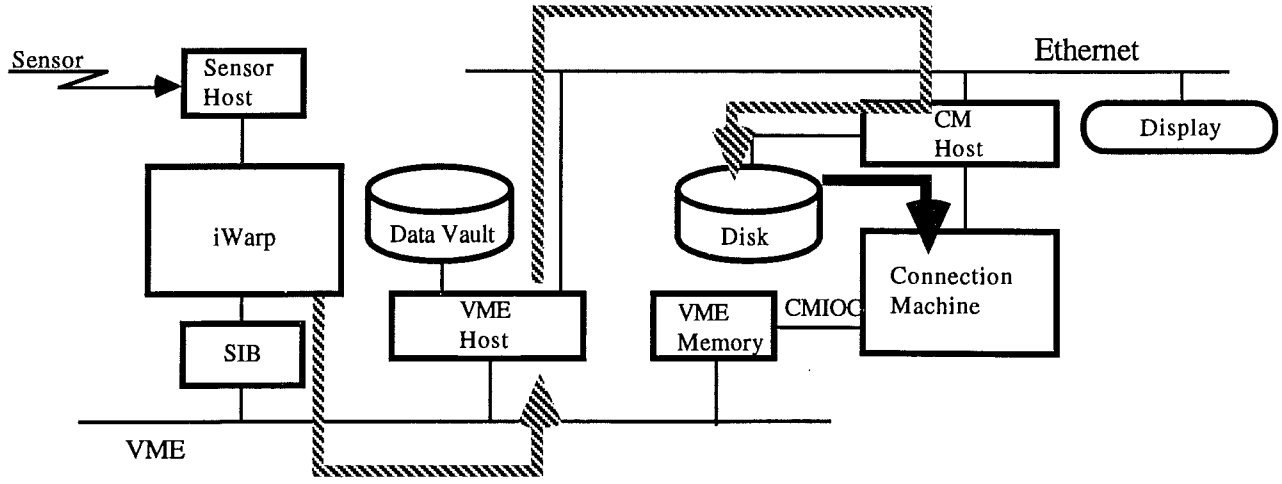


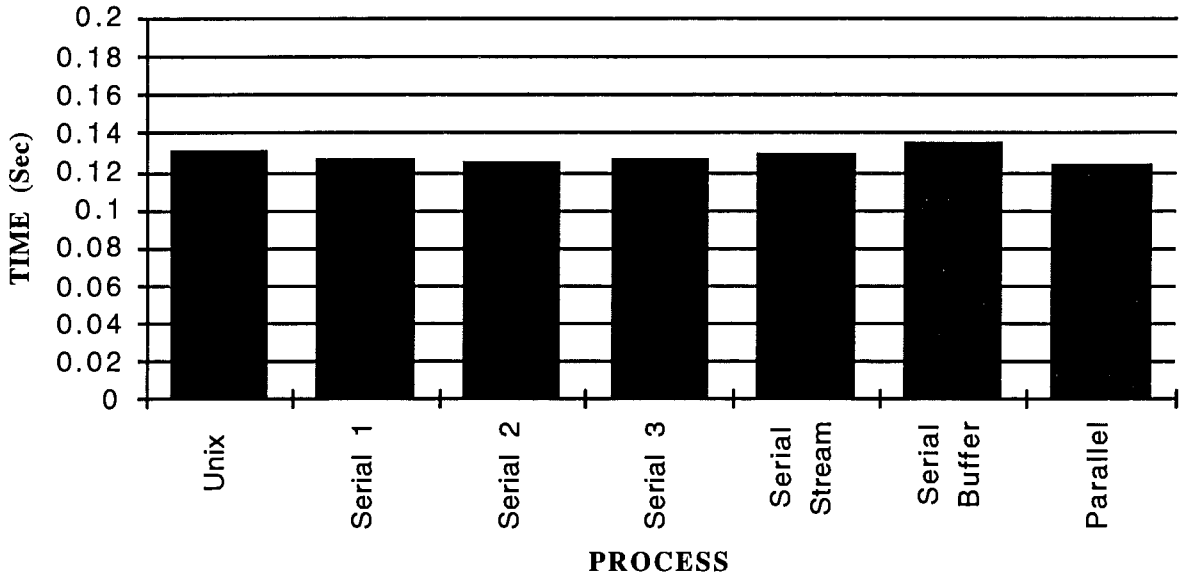
Figure 3c - Data Flow via Unix Disk

Figure 4 provides the results of a study which compares the various command alternatives described above for a 131 kbyte data block. This size represents 256 frequency bins x 64 beams x 1 time step which, at present, is considered to be the optimum data block size for the ASPA project. The first bar in the graph shows the time required to read a Unix file from the CM host disk (Figure 3c). The remaining bars represent data vault transfer times (Figure 3a). The exact command suite associated with each bar is described below.

- Unix - Unix read,
  CM_write_to_news_array
- Serial 1 - CMFS_read,
  CMFS_transpose_always,
  CMFS_cm_to_standard_byte_order
- Serial 2 - CMFS_read_always,
  CMFS_transpose_always,
  CMFS_cm_to_standard_byte_order
- Serial 3 - CMFS_read_always,
  CMFS_twuffle_from_serial_order_always,
  CMFS_cm_to_standard_byte_order
- Serial Stream - CMFS_fcntl,
  CMFS_streaming_iostat,
  CMFS_partial_read_file_always,
  CMFS_twuffle_from_serial_order_always,
  CMFS_cm_to_standard_byte_order
- Serial Buffer - CMFS_buffered_read_file_always,
  CMFS_flush,
  CMFS_twuffle_from_serial_order_always,
  CMFS_cm_to_standard_byte_order
- Parallel - CMFS_read_always

As Figure 4 demonstrates, the time required to perform any of the data transfer options does not vary significantly from option to option. With a mean transfer time of 0.128 seconds across options, the greatest disparity between it and any of the observed values is 0.007 seconds. The variance is 1.38×10^{-5}. Contrary to initial predictions, these results imply that the iWarp to CM interface could be designed independent of the data transfer process, at least from a CM point of view. It was originally anticipated that the streaming I/O and parallel read processes would provide the fastest times; streaming I/O because it minimizes overhead particularly for large data transfers and parallel read because it eliminates the data format anomalies associated with serial data. This, however, was not found to be the case. Reformatting the data so that the CM can use it appears to take little time which is evidenced by comparing the parallel read time with that of any of the serial read times. Parallel reads do not require the data to be reformatted. And although the Connection Machine File System (CMFS) library provides several different commands for reformatting data, it does not appear that any specific command suite provides significantly better performance than any other.

Figure 4 - Process Times for 131 KByte Transfer



In addition to investigating the impact of various data transfer processes, the impact of data block size was also investigated. Figure 5 compares the amount of time it takes to transfer varying amounts of data using the same serial command suite, namely "Serial 3" described above. From the figure, it can be seen that as the size increases in a linear manner from 131KB to 1MB, so does transfer time. Parallel read times were also evaluated at the extremities and also appear to increase in a linear fashion. For larger data blocks, parallel reads provide slightly better times than do serial reads. Figure 6 compares throughput rates against size. When compared in this manner, size has little impact on throughput.

The results of this study suggest that there exists a great deal of freedom available with the design of the iWarp to CM interface. The different command suites available for transferring data into the CM provide similar performance while data transfers sizes provide similar throughput rates. A slight advantage exists when transferring large parallel data blocks over serial ones. However, this study does not take into account the cost of transferring data from the iWarp to an intermediate storage location. This could potentially alter the conclusions stated herein. Also, another factor contributing to the uniform results could be disk access time. Each of the processes investigated required access to disk. This access time could be, and probably is, slower than any of the communication mechanisms investigated, thereby biasing the results. This too will be investigated further.

Figure 5 - Transfer Time vs Size

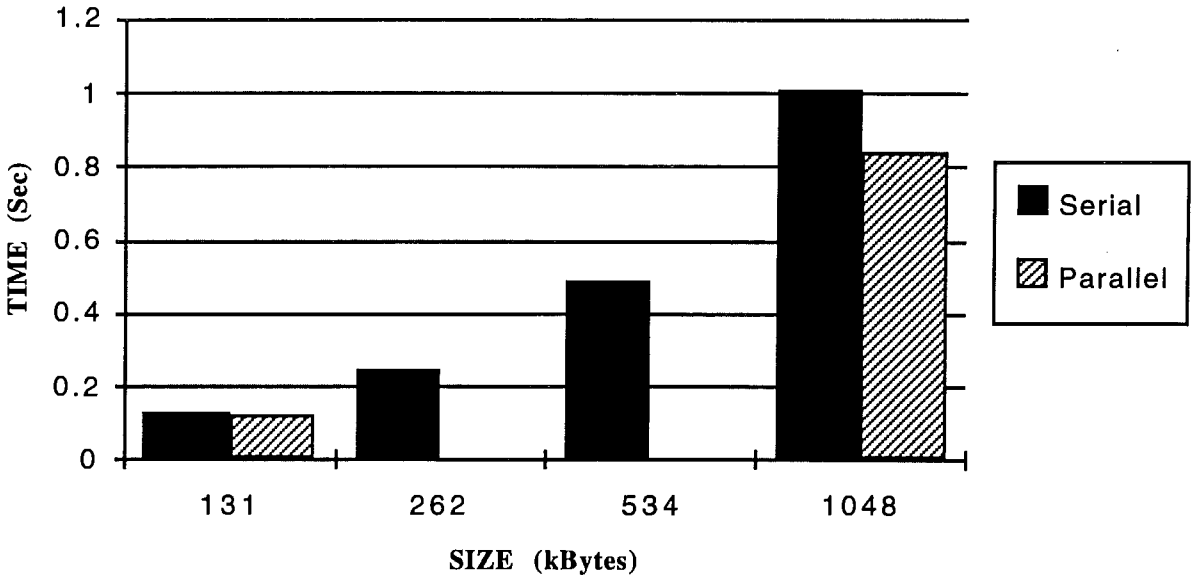
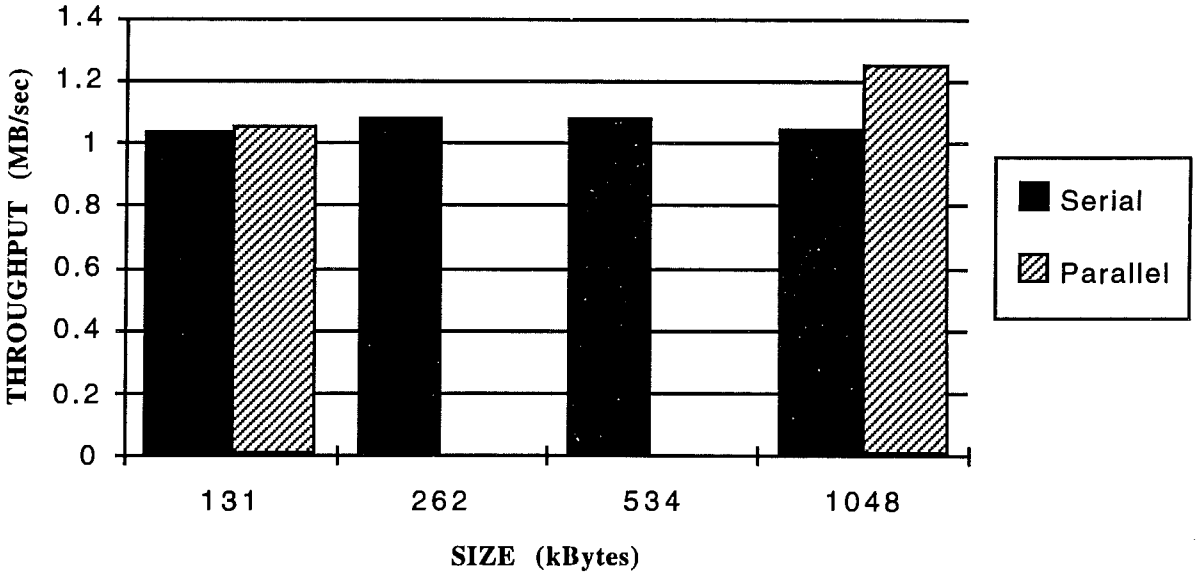


Figure 6 - Throughput vs Size



4.0 CORNER TURNING STUDY

One of the artifacts of using two architecturally different computers is that they often require data to be organized and processed differently in order to take advantage of the computer's unique architecture. This being the case for the ASPA system, incompatibilities exist in the way the data is laid out for processing in the iWarp and CM. This data, when transferred between machines, must be re-organized. Figure 7 describes the data layouts: data on the iWarp is organized in a three dimensional space comprised of: n time series x m frequency bins x k beams; data on the CM is organized in a two dimensional space comprised of k beams x m frequency bins for a single time step. Before iWarp data can be processed on a CM, the data must be corner turned where a single times series is extracted from the three dimensional space and formatted for frequency and beams. The goal of this study is to determine where the corner turning process is to take place, on either the VME host or CM. A third possibility would have been on the iWarp, but this option was not investigated since the iWarp is currently resource limited. One of the contributing factors in affecting the corner turn performance is the actual size of the data processed. Number of beams, frequency bins, and time were varied in order to assess its impact.

Figures 8 and 9 show the CPU time required to corner turn data with time series equal to 8 and 12 respectively. Number of frequency bins and beams where varied such that they ranged from 64 to 2048 in increments of power of two (which was done in order to comply with the CM architecture restrictions mentioned above). The size, which is represented along the 'X' axis, represents the total amount of data corner turned and was calculated by multiplying the number of frequency bins by beams by time series by 8 bytes (complex data type). Several size combinations of frequency bins and beams could exist for producing an equivalent total size. As an example:

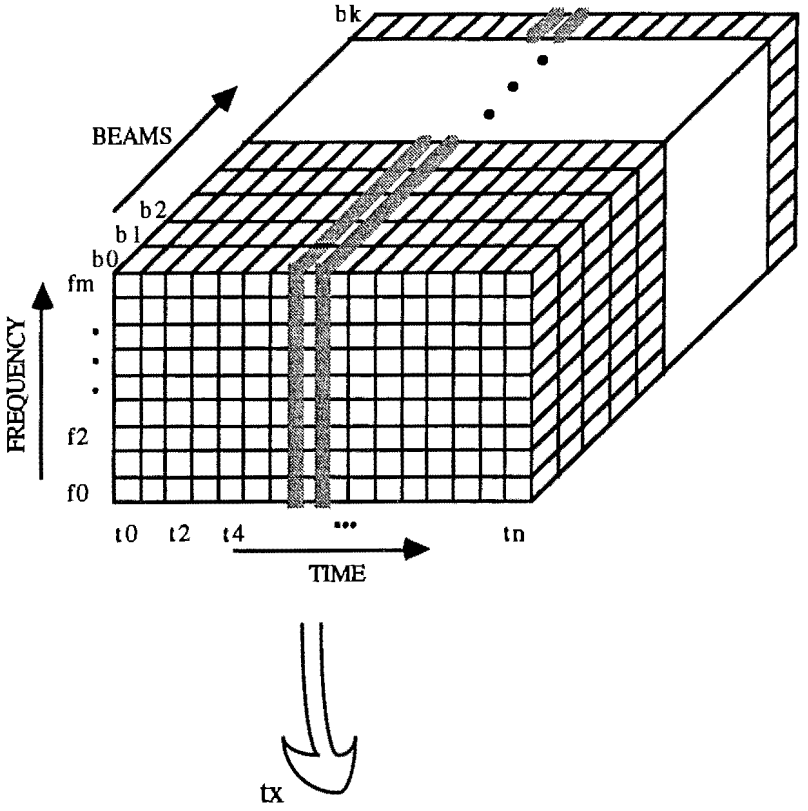
frequency bins = 128 beams = 64 time series = 8
==> 128 x 64 x 8 x 8 bytes = 0.5 MBytes

frequency bins = 64 beams = 128 time series = 8
==> 64 x 128 x 8 x 8 bytes = 0.5 MBytes

For this study, each possible combination of frequency bins and beam sizes were evaluated. It was found that the amount of CPU time required to corner turn was independent of the number of frequency bins and beam sizes; it is however, dependent on the total amount of data corner turned. For the example above, it takes the same amount of time to corner turn 128 frequency bins by 64 beams as it does 64 frequency bins by 128 beams. The table at the bottom of Figures 8 and 9 show all possible combinations for a specific total size.

As can be seen from each of the graphs, it takes substantially more time to perform the corner turn on the VME host (a Sun 4/ 360 workstation) than it does on the CM. Based on these results as well as other ASPA timing requirements, only data sizes of 0.5 MBytes and less can be processed on the VME host while the CM is capable of processing sizes up to 32 MBytes. If the total size is under 4 MBytes, the amount of time to process the data on the CM is insignificant. It should be noted that for data sizes under 0.5 MBytes, the data had to be mapped to larger data structures in order to be processed by the CM. This is the result of an architecture restriction which requires that CM data structures meet certain size constraints. The effects of this are shown in Figures 8 and 9 with increased processing times for 0.25 and 0.37 MBytes respectively.

iWarp Data Layout



CM Data Layout

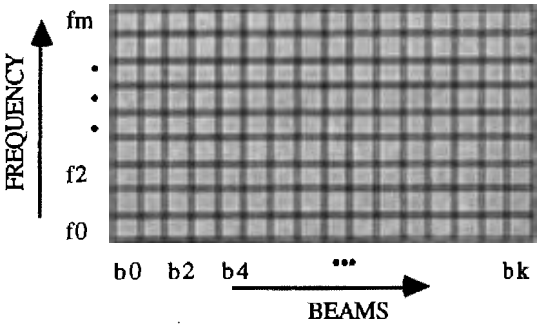
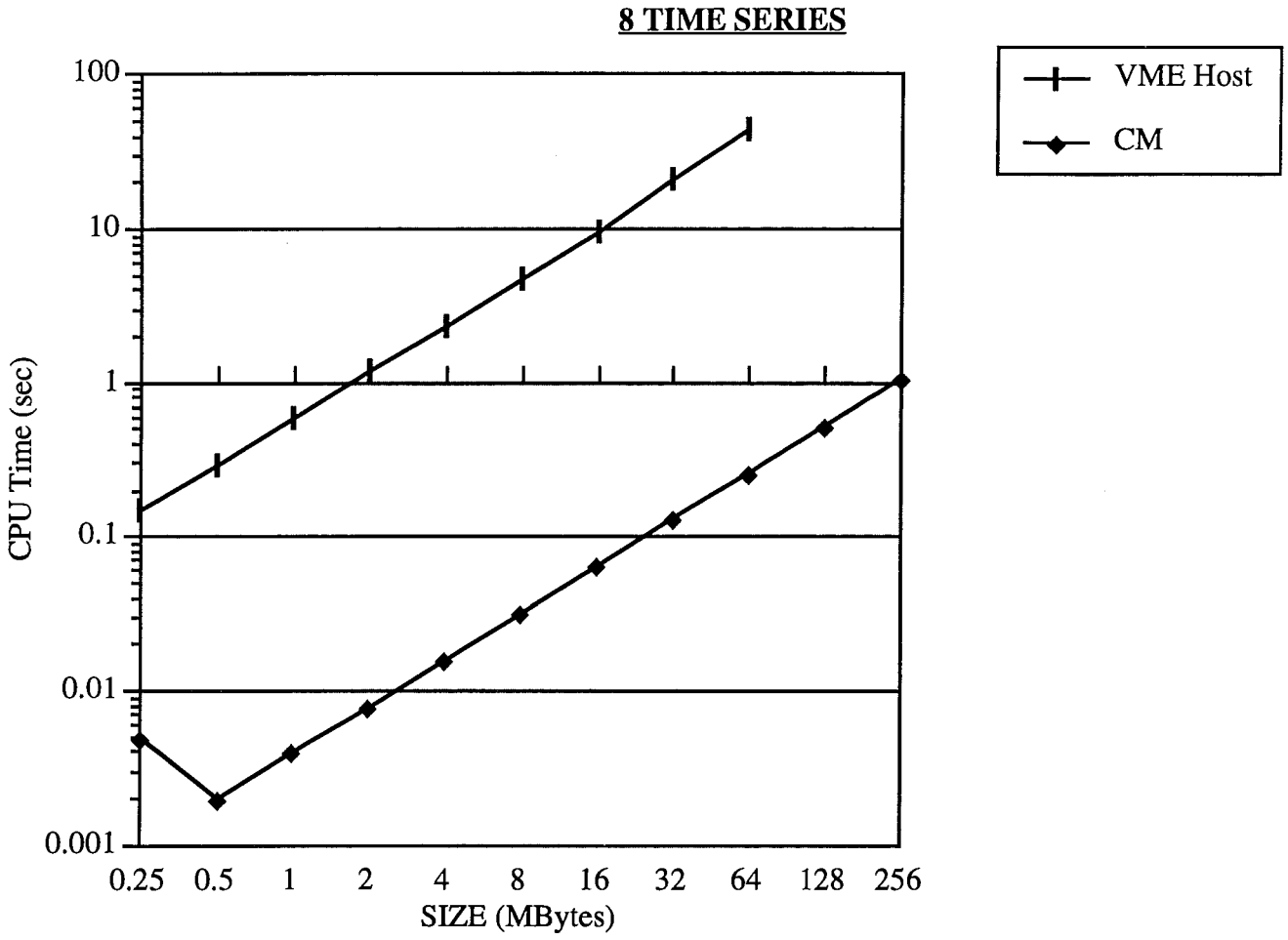


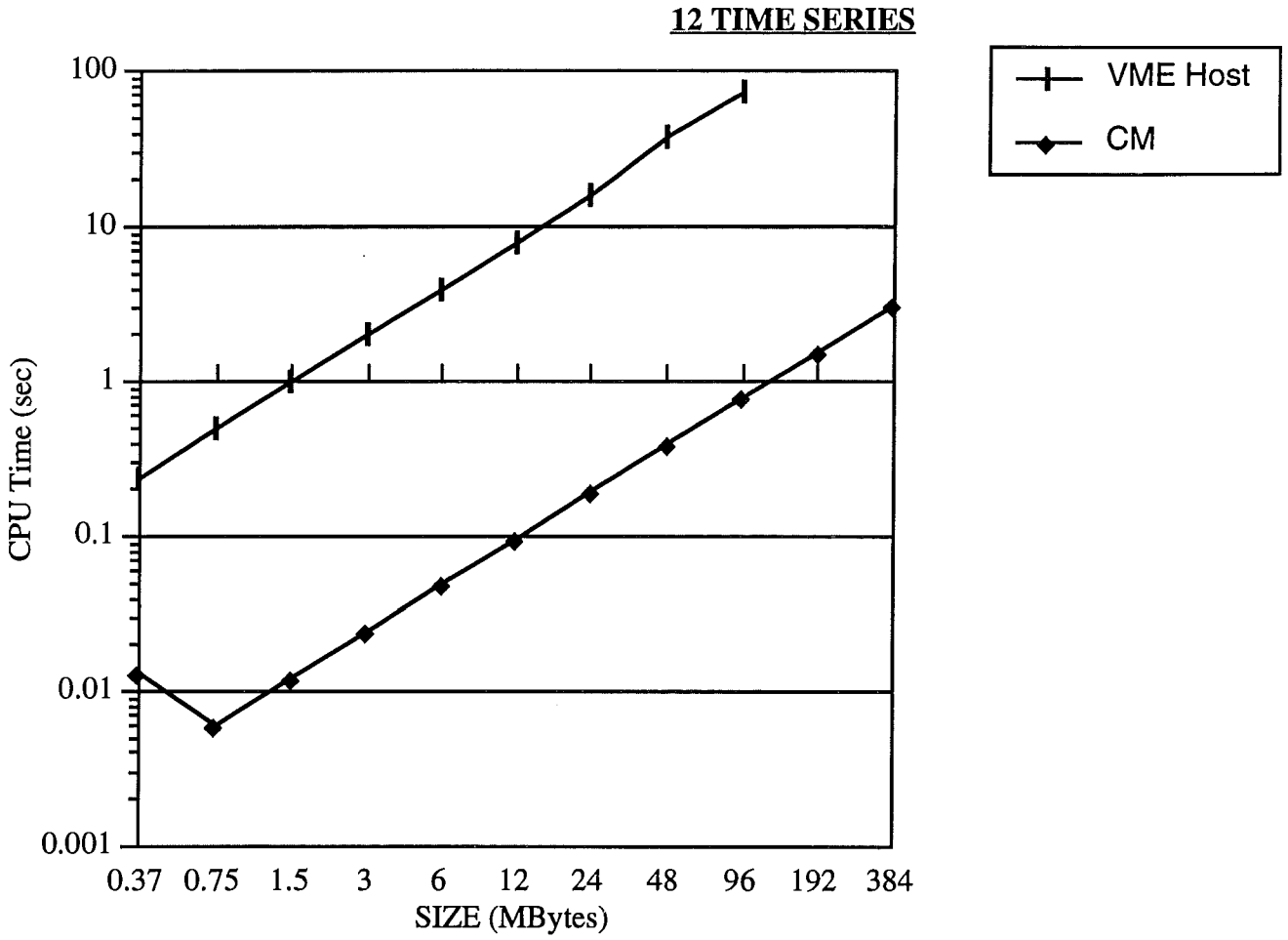
Figure 7 - Corner Turning

Figure 8 - Corner Turn Timing Results



SIZE TO SPACIAL RELATIONSHIP	
Size	= Frequency Bins x Beams x 8 Time Series x 8 Bytes
0.25MB	= 64 x 64
0.50MB	= 128 x 64 I 64 x 128
1MB	= 256 x 64 I 128 x 128 I 64 x 256
2MB	= 512 x 64 I 256 x 128 I 128 x 256 I 64 x 512
4MB	= 1024 x 64 I 512 x 128 I 256 x 256 I 128 x 512 I 64 x 512
8MB	= 2048 x 64 I 1024 x 128 I 512 x 256 I 256 x 512 I 128 x 1024 I 64 x 2048
16MB	= 2048 x 128 I 1024 x 256 I 512 x 512 I 256 x 1024 I 128 x 2048
32MB	= 2048 x 256 I 1024 x 512 I 512 x 1024 I 256 x 2048
64MB	= 2048 x 512 I 1024 x 1024 I 512 x 2048
128MB	= 2048 x 1024 I 1024 x 2048
256MB	= 2048 x 2048

Figure 9 - Corner Turn Timing Results



SIZE TO SPACIAL RELATIONSHIP

Size = Frequency Bins x Beams x 12 Time Series x 8 Bytes

0.37MB	= 64 x 64
0.75MB	= 128 x 64 I 64 x 128
1.5MB	= 256 x 64 I 128 x 128 I 64 x 256
3MB	= 512 x 64 I 256 x 128 I 128 x 256 I 64 x 512
6MB	= 1024 x 64 I 512 x 128 I 256 x 256 I 128 x 512 I 64 x 512
12MB	= 2048 x 64 I 1024 x 128 I 512 x 256 I 256 x 512 I 128 x 1024 I 64 x 2048
24MB	= 2048 x 128 I 1024 x 256 I 512 x 512 I 256 x 1024 I 128 x 2048
48MB	= 2048 x 256 I 1024 x 512 I 512 x 1024 I 256 x 2048
96MB	= 2048 x 512 I 1024 x 1024 I 512 x 2048
192MB	= 2048 x 1024 I 1024 x 2048
384MB	= 2048 x 2048

5.0 CONCLUSIONS

The two studies described herein provided substantial insight into the data transfer process between the iWarp and CM. By examining the performance behavior of the CM and its supporting environment, a number of conclusions can be drawn about the design of the iWarp/CM interface. Although the CM provides a variety of mechanisms for data input, the overall performance of each one varies little. This provides a great deal of flexibility in the way data can be sent to the CM thereby increasing the number of design options. Similar conclusions can be drawn over the size of the data transferred. Within the scope of the transfer sizes analyzed, throughput rates varied little providing additional design flexibility. However, this flexibility is made available at the expense of using disk as an intermediate storage location. With relatively slower access times, disk access was found to bias communication time although it is not apparent by how much due to the limited scope of this study. The Corner Turning study demonstrated that the CM is more than capable of performing the corner turn process particularly when the amount of data is kept under 4 MBytes. Given ASPA timing requirements as well as capacity limitations, this number can be extended to 32 MBytes. The time it takes to corner turn data is dependent on the total amount of data processed and is affected little by the number of elements along a dimension. However, CM architecture restrictions require the number of elements along any dimension to be a power of two which can led to extremely inefficient use of the CM.

Given these results, each of the design options highlighted in Figures 3a-c are still viable; however, it can be anticipated that when iWarp I/O is accounted for, this will change. In general, writing to disk can be a very expensive process and should be avoided particularly when maximum throughput is of utmost concern. This being the case, then transferring data directly via VME memory (Figure 3b) should provide the optimal interface. If, however, it is necessary to store the data on disk before the transfer to the CM is made, it is likely the data vault will provide the best times. This is due to a more direct path between machines. Additional studies need to be conducted in order to confirm these conclusions.

APPENDIX I
COMPUTER DESCRIPTION

Connection Machine CM200-a

Manufacturer:	Thinking Machines Corp.
Architecture:	Single Instruction Multiple Data (SIMD)
Processor type:	CM-2 processor
Number of processors:	8192 with one 64-bit Floating Point Processor per 32 CM-2 processors
Word Size:	1 bit
Memory per processor:	1 MBit
Total Memory:	1 GByte
Number I/O channels:	3 per system: 2 CMIOC + 1 pBus
I/O throughput:	50 MBytes/sec per CMIOC; 5 MBytes/sec per pBus
System performance:	2.3 GFLOPS

64-cell iWarp

Manufacturer:	Intel Corp.
Architecture:	Multiple Instruction Multiple Data (MIMD)
Processor type:	iWarp cell
Number of processors:	64
Word Size:	32 bits
Memory per processor:	2 MBytes
Total memory:	128 MBytes
Number I/O channels:	2
I/O throughput:	18 MBytes/sec per channel
System performance:	1.2 GFLOPS

DISTRIBUTION LIST

Internal Distribution

Code	
0261	(NL Library (2))
0262	(NPT Library (2))
2094	(J. DePrimo)
21	(W. Coggins)
2121	(W. Fischer, M. Schindler)
2122	(J. O'Sullivan)
2123	(G. Bowman, R. Choma, S. Dzerovych, J. Ianniello, J. Law M. Maguire, N. Owsley, T. Tetlow)
2133	(H. Schloemer)
2141	(P. Davis)
2142	(D. Abraham)
2143	(J. Marsh)
215	(H. Watt)
2151	(T. Choinski, D. DaRos, D. Organ)
2152	(T. Anderson, I. Ferber, R. Latourette)
2153	(W. Bernecky, A. Edmonds, S. Harrison, R. Howbrigg, J. Ionata B. Iwatake, L. Karasevich, M. Krzych (5), J. Munoz)
2154	(G. O'Brien)
2191	(R. Murdock)

External Distribution

K. Bromley
NRAD
San Diego, CA

T. DeYoung
ARPA
3701 N. Fairfax Dr.
Arlington, VA

B. Wald
ONR
800 N. Quincy St.
Arlington, VA

B. Wasilauski
NRAD
San Diego, CA

Total: 47